

UNIT 3

Software Estimation Techniques

(7 Hours) — Software Project Management (CAC407)

NOTE: These notes may contain errors. Please verify important points, definitions and calculations with your textbook and teacher before relying on them.

3.1 Software Effort Estimation

Estimation is the process of predicting the effort (person-months), duration and cost needed to complete a software project. Estimates are needed for bidding, budgeting, scheduling, staffing and deciding whether a project is worth doing. An estimate is a *prediction*, not a promise, and always has uncertainty.

Typical quantities estimated are **size** (lines of code, function points), **effort** (person-months), **duration** (calendar time) and **cost** (money).

3.2 Problems with Over- and Under-estimation

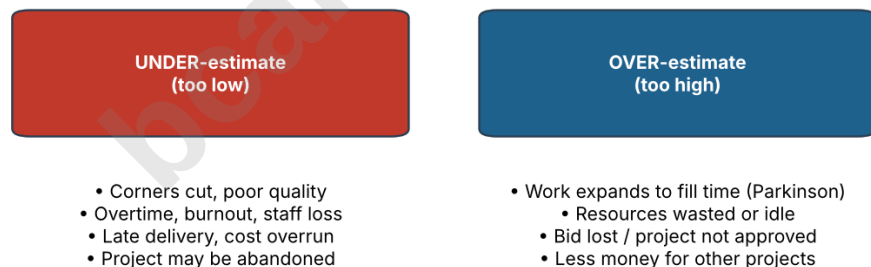


Figure 3.1 — Consequences of poor estimates

Under-estimation is generally more dangerous, because the project starts under pressure and quality suffers. However, over-estimation is also harmful: according to **Parkinson's Law**, work expands to fill the time available.

3.2.1 Why estimation is difficult

- Software is intangible and each project is different (little history to compare).
- Requirements are incomplete and change during the project.

- Productivity varies greatly between individuals and teams.
- Subjective factors: optimism, political pressure to give lower numbers, 'anchoring' on earlier figures.
- Technology, tools and reused components are uncertain.

3.2.2 Cone of uncertainty

Estimates become more accurate as the project progresses and more is known. The range of error is widest at the start.

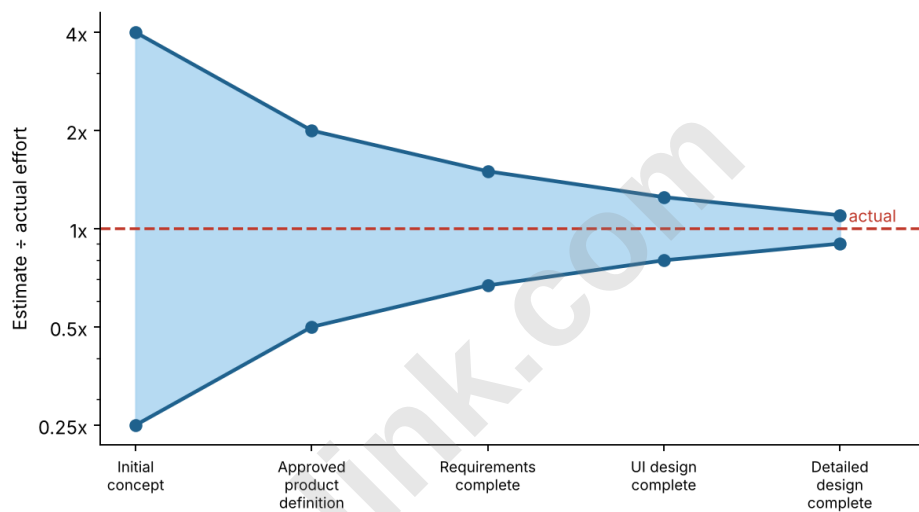


Figure 3.2 — Cone of uncertainty (typical ranges after McConnell)

3.3 Basis of Software Estimating

- **Historical data** — records of size, effort and duration from earlier projects of the organization.
- **Measure of work (size)** — lines of code (LOC/KLOC), function points, object points, use-case points.
- **Complexity factors** — product, platform, personnel and project attributes.
- **Productivity rate** — output per unit effort (for example, LOC per person-month, FP per person-month).
- **Effort = Size ÷ Productivity** is the basic relationship used by many methods.

3.3.1 Size measures

Measure	Description	Strength / weakness
LOC / KLOC	Count of source lines (thousands).	Easy to count after coding, but depends on language and style; hard to predict early.
Function Points (FP)	Size from functionality delivered to the user.	Language independent, available early from requirements; counting needs training.
Use-case points	Size from use cases and actors.	Useful for object-oriented analysis.
Story points	Relative size of user stories (agile).	Quick, team-calibrated; not comparable across teams.

3.4 Software Effort Estimation Techniques

Category	Techniques
Expert based	Expert judgment, Delphi technique
Analogy based	Estimating by analogy (case-based)
Decomposition based	Bottom-up estimating, work-breakdown based
Top-down	Top-down approach with allocation of total effort
Algorithmic / parametric models	COCOMO, Function Point model, regression models

3.4.1 Expert judgment

One or more experienced people estimate the effort from their knowledge of similar work. It is quick and can consider special factors, but it depends on the individual's bias and memory.

- **Delphi technique** — several experts estimate independently and anonymously; a coordinator shows the summary; experts revise over a few rounds until estimates converge. This reduces dominance by one person.
- **Wideband Delphi** adds a group discussion between rounds.
- Three-point estimates (optimistic, most likely, pessimistic) improve expert judgment: $E = (o + 4m + p) / 6$.

3.4.2 Estimating by analogy

The estimator finds one or more completed projects that are similar (**source cases**), and adjusts their actual effort for the differences to estimate the new project (**target**). It needs a good project history.

Example 3.1 — Analogy

A completed project (a library system) had size 20 KLOC and took 60 person-months. A new project (a college information system) is expected to be about 30 KLOC with similar team and technology. Estimate its effort.

Solution

Productivity of old project = $20 \text{ KLOC} / 60 \text{ pm} = 0.333 \text{ KLOC per pm}$ (3 pm per KLOC).

New effort $\approx 30 \times 3 = \mathbf{90 \text{ person-months}}$.

If the new team is less experienced, a further adjustment (say +10%) could be applied → about 99 person-months.

3.4.3 Bottom-up estimating

The work is broken down (WBS) to small components; each is estimated separately and the results are **summed**, then management and integration effort are added. It is detailed and usually more accurate, but needs the design to be known and takes more time. It can miss system-level activities (integration, documentation, management).

Example 3.2 — Bottom-up

Module estimates (person-days): Login 10, Student form 25, Fee payment 30, Reports 20, Admin panel 15.

Integration and system testing is estimated at 20% of the coding total, and project management at 10% of the grand total before management.

Solution

Sum of modules = $10 + 25 + 30 + 20 + 15 = 100 \text{ person-days}$.

Integration/testing = 20% of 100 = 20. Subtotal = 120.

Management = 10% of 120 = 12.

Total = 132 person-days.

3.4.4 Top-down approach

An overall estimate for the whole project is made first (from a parametric model, analogy or expert opinion), then it is divided among the components or activities, usually by percentages from past projects. It is fast and considers system-level work, but may miss technical details of individual components.

Activity	Typical share of effort
Requirements and analysis	10–15%
Design	15–20%
Coding and unit test	30–35%
Integration and system test	20–25%
Documentation, management, other	10–15%

Note: These percentages vary by organization and project type; use your own historical data when available.

3.4.5 Bottom-up vs top-down

Aspect	Top-down	Bottom-up
Start from	Whole project	Individual components/tasks
Needs	Overall size or analogous project	Detailed breakdown (WBS), design information
Effort to produce	Low, fast	High, slow
Strengths	Considers system-level work; early use	Detailed, usually more accurate, gives ownership to workers
Weaknesses	May overlook technical difficulty of parts	May miss integration and management work

3.5 Parametric (Algorithmic) Models

Parametric models use mathematical formulas relating effort to size and other factors: **Effort = $a \times (\text{Size})^b \times (\text{adjustment factors})$** . The constants are calibrated from historical data.

3.5.1 Basic COCOMO (Boehm, 1981)

The **Constructive Cost Model** predicts effort and schedule from size in KLOC. Projects are classified into three modes:

Mode	Description	a	b	c	d
Organic	Small team, familiar problem, relaxed constraints	2.4	1.05	2.5	0.38
Semi-detached	Medium team, mixed experience and requirements	3.0	1.12	2.5	0.35

Mode	Description	a	b	c	d
Embedded	Tight hardware/operational constraints, complex	3.6	1.20	2.5	0.32

- **Effort E** = $a \times (\text{KLOC})^b$ person-months
- **Development time D** = $c \times (E)^d$ months
- **Average staff N** = E / D persons

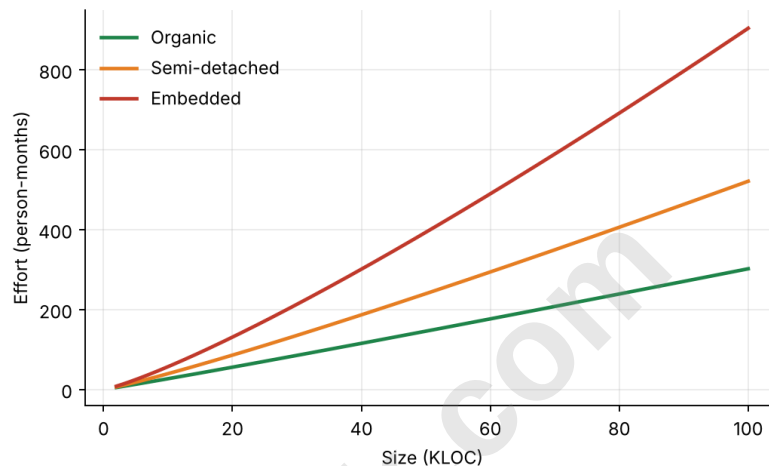


Figure 3.3 — Basic COCOMO effort versus size for the three modes

Example 3.3 — Basic COCOMO

A project is estimated at 32 KLOC. Compute effort, time and staff for the organic and the embedded modes.

Solution

Organic: $E = 2.4 \times 32^{1.05} = 91.3 \text{ pm}$; $D = 2.5 \times 91.3^{0.38} = 13.9 \text{ months}$; $N = 91.3 / 13.9 = 6.6 \text{ persons}$.

Embedded: $E = 3.6 \times 32^{1.20} = 230.4 \text{ pm}$; $D = 2.5 \times 230.4^{0.32} = 14.3 \text{ months}$; $N = 16.2 \text{ persons}$.

The embedded mode needs much more effort for the same size because of the stricter constraints.

Note: Values are rounded; small differences from a calculator are normal.

3.5.2 Intermediate and Detailed COCOMO; COCOMO II

- **Intermediate COCOMO** multiplies the nominal effort by an Effort Adjustment Factor (EAF) from 15 cost drivers (product, hardware, personnel, project attributes). $\text{Effort} = a \times (\text{KLOC})^b \times \text{EAF}$.

- **Detailed COCOMO** applies cost drivers separately at each phase.
- **COCOMO II** (Boehm, 2000) supports modern development: Application Composition (object points), Early Design (function points) and Post-Architecture (source lines) models, with scale factors replacing the three modes.

3.5.3 Function Point analysis (Albrecht)

Function Points measure size from the functions the user sees. The steps are:

- Count five component types: External Inputs (EI), External Outputs (EO), External Inquiries (EQ), Internal Logical Files (ILF) and External Interface Files (EIF).
- Rate each as low, average or high complexity and multiply by its weight to get **Unadjusted FP (UFP)**.
- Rate 14 General System Characteristics from 0 to 5 and compute the **Value Adjustment Factor VAF** $\text{VAF} = 0.65 + 0.01 \times \Sigma F_i$.
- **FP** $\text{FP} = \text{UFP} \times \text{VAF}$; Effort = FP ÷ productivity (FP per person-month).

Component	Low	Average	High
External Input (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiry (EQ)	3	4	6
Internal Logical File (ILF)	7	10	15
External Interface File (EIF)	5	7	10

Example 3.4 — Function points

A system has 20 EI, 15 EO, 10 EQ, 8 ILF and 4 EIF, all of average complexity. The sum of the 14 characteristic ratings is 42. Team productivity is 10 FP per person-month. Find FP and effort.

Solution

$$\text{UFP} = 20 \times 4 + 15 \times 5 + 10 \times 4 + 8 \times 10 + 4 \times 7 = 80 + 75 + 40 + 80 + 28 = \mathbf{303}.$$

$$\text{VAF} = 0.65 + 0.01 \times 42 = \mathbf{1.07}.$$

$$\text{FP} = 303 \times 1.07 = \mathbf{324.2}.$$

$$\text{Effort} = 324.2 / 10 = \mathbf{32.4 \text{ person-months}}.$$

3.6 Comparison and Choosing a Technique

Technique	Advantages	Limitations
Expert judgment	Fast, flexible	Subjective, biased
Analogy	Based on real projects	Needs a good, comparable history
Bottom-up	Detailed, accurate	Needs full breakdown; time-consuming
Top-down	Quick, early	May miss technical details
Parametric (COCOMO, FP)	Repeatable, objective	Needs calibration; size must be estimated first

Good practice is to use **more than one technique** and compare the results. If they differ widely, investigate why, and record assumptions. Re-estimate at each major milestone and keep actual data to improve future estimates.

Key Takeaways

- ✓ Estimates predict size, effort, duration and cost; accuracy improves as the project progresses (cone of uncertainty).
- ✓ Under-estimation causes pressure and poor quality; over-estimation wastes resources (Parkinson's Law).
- ✓ Expert judgment (Delphi), analogy, bottom-up, top-down and parametric models are the main techniques.
- ✓ Basic COCOMO: $E = a \cdot K \text{LOC}^b$, $D = c \cdot E^d$, $N = E/D$, with organic, semi-detached and embedded modes.
- ✓ Function Points: UFP from weighted counts, $\text{VAF} = 0.65 + 0.01 \sum F_i$, $\text{FP} = \text{UFP} \times \text{VAF}$.
- ✓ Use several techniques, record assumptions and re-estimate regularly.

— End of Unit 3 —